

Programação Orientada a Objetos

CLASSES ABSTRATAS E INTERFACES

Turmas B e C

Nádia Félix

(nadia.felix@ufg.br)

Dirson S. Campos

(dirson_campos@ufg.br)

17/02/2022

Programação Orientada a Objetos

Classes Abstratas

- Uma classe abstrata é uma classe que não pode ser instanciada ... não pode ser diretamente utilizada para criar objetos;
- O propósito de criação de uma classe abstrata é de fornecer uma superclasse apropriada para que outras classes utilizem como base (herança);
- Seus métodos podem ter implementação ou podem ser abstratos também (sem implementação).

Programação Orientada a Objetos

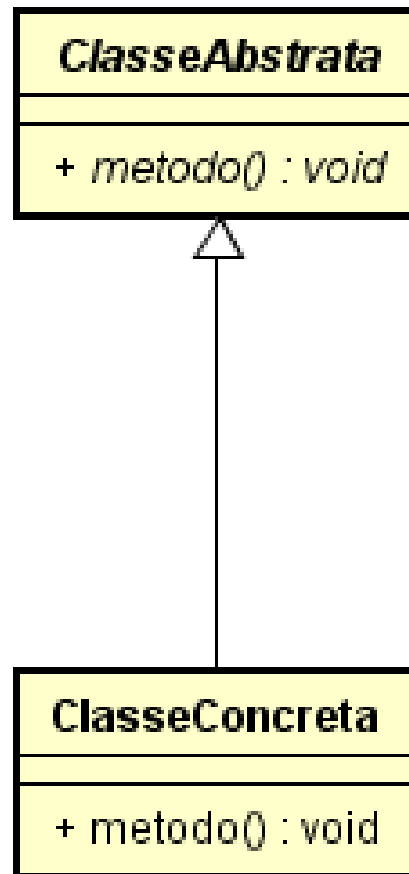
Classes Abstratas (Java)

```
public abstract class ClasseAbstrata {  
    public abstract void metodo( );  
}
```

```
public class ClasseConcreta extends ClasseAbstrata {  
    public void metodo( ) {  
        System.out.println("Isto Funciona!");  
    }  
}
```

Programação Orientada a Objetos

Classes Abstratas (UML)

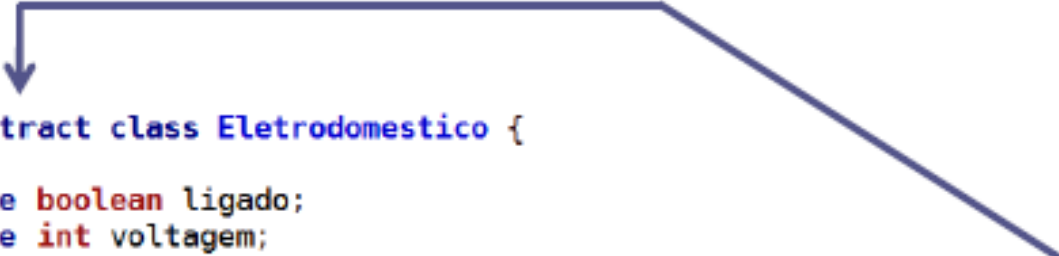


Classe e método abstrato

Classe e método concreto

Programação Orientada a Objetos

Classes Abstratas em Java



```
public abstract class Eletrodomestico {  
    private boolean ligado;  
    private int voltagem;  
  
    public Eletrodomestico(boolean ligado, int voltagem) {  
        this.ligado = ligado;  
        this.voltagem = voltagem;  
    }  
  
    public void setVoltagem(int voltagem) {  
        this.voltagem = voltagem;  
    }  
  
    public int getVoltagem() {  
        return this.voltagem;  
    }  
  
    public void setLigado(boolean ligado) {  
        this.ligado = ligado;  
    }  
  
    public boolean isLigado() {  
        return ligado;  
    }  
}
```

A palavra reservada **abstract** indica que a classe é abstrata e portanto não pode ser instanciada diretamente (precisa ser herdada)

Programação Orientada a Objetos

Métodos abstratos

- Uma classe abstrata pode conter **métodos abstratos** ... ou seja ... métodos sem implementação na classe abstrata que exigem a implementação nas subclasses
- A técnica de especificar métodos abstratos permite que o projetista decida quais são os comportamentos que as subclasses devem ter ... mas sem determinar **como** tais comportamentos serão implementados ...
- Somente classes abstratas podem ter métodos abstratos.

Programação Orientada a Objetos

Métodos abstratos em Java

```
public abstract class Eletrodomestico {  
    private boolean ligado;  
    private int voltagem;  
  
    public Eletrodomestico(boolean ligado, int voltagem) {  
        this.ligado = ligado;  
        this.voltagem = voltagem;  
    }  
  
    public abstract void ligar();  
    public abstract void desligar();  
  
    public void setVoltagem(int voltagem) {  
        this.voltagem = voltagem;  
    }  
  
    public int getVoltagem() {  
        return this.voltagem;  
    }  
  
    public void setLigado(boolean ligado) {  
        this.ligado = ligado;  
    }  
  
    public boolean isLigado() {  
        return ligado;  
    }  
}
```

Métodos abstratos não podem ter implementação na classe abstrata.

Nas subclasses os métodos abstratos devem ser obrigatoriamente implementados.

```
class TV extends Eletrodomestico {  
    private int tamanho;  
    private int canal;  
    private int volume;  
  
    public TV(int tamanho, int voltagem) {  
        super(false, voltagem);  
        this.tamanho = tamanho;  
        this.canal = 0;  
        this.volume = 0;  
    }  
  
    public void desligar() {  
        super.setLigado(false);  
        setCanal(0);  
        setVolume(0);  
    }  
  
    public void ligar() {  
        super.setLigado(true);  
        setCanal(3);  
        setVolume(25);  
    }  
  
    //Demais métodos  
}
```

Programação Orientada a Objetos

Exemplo Banco 1

Representação de contas correntes e contas poupanças em um banco.

Toda Conta de um banco tem um número, um titular (nome e cpf) e saldo.

As contas correntes guardam também a informação do limite.

As contas poupanças guardam a informação da taxa de juros.

Existe uma regra para a validação dos números das contas:

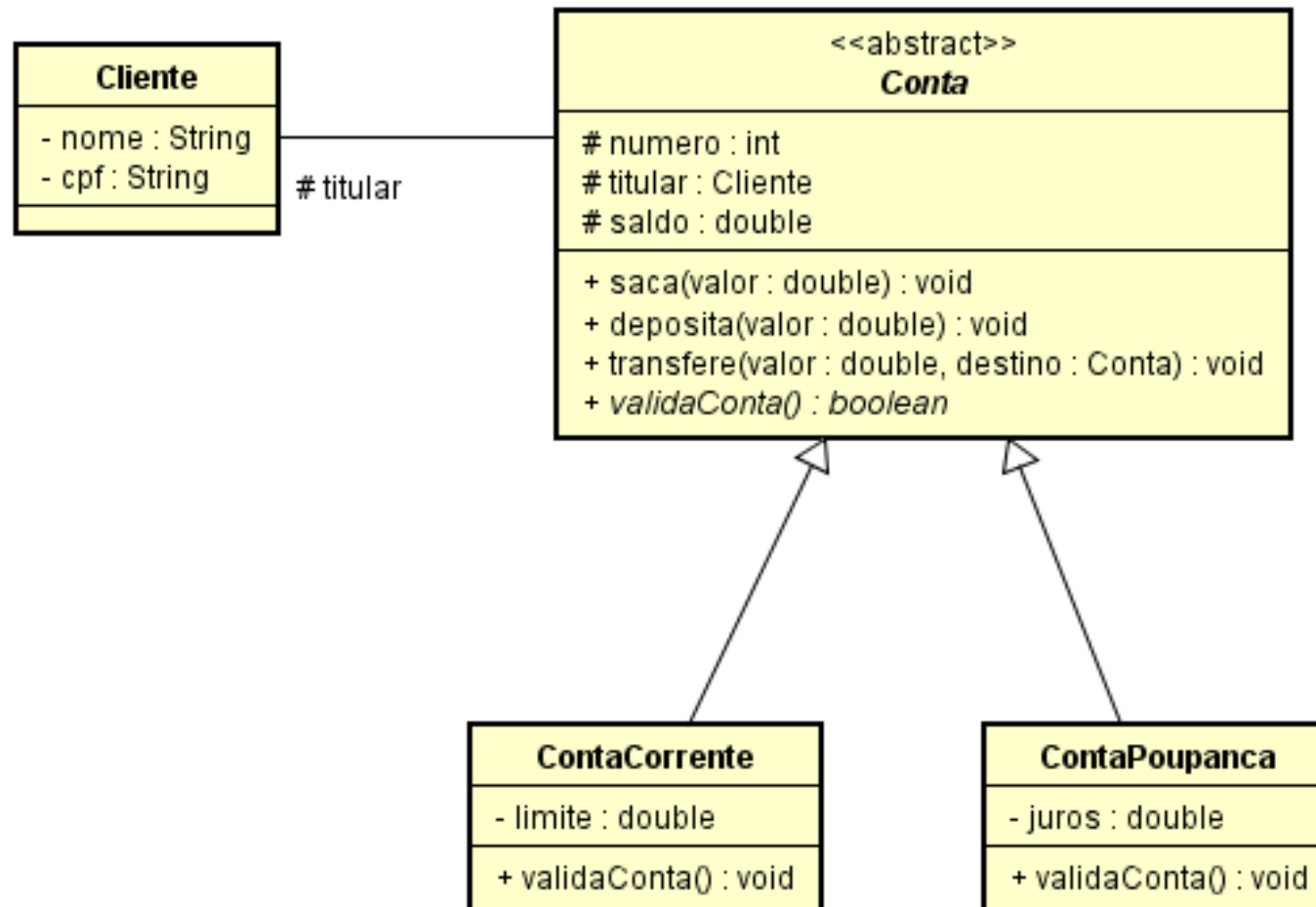
- As contas correntes são números maiores que 100000
- As contas poupanças são números menores ou iguais a 100000

Toda conta pode fazer as operações de sacar, depositar e transferir.

Deve haver uma operação para validar as contas com relação à regra dos números.

Programação Orientada a Objetos

Exemplo Banco 1 (UML)



Programação Orientada a Objetos

Exemplo Banco 1 (Java)

```
public abstract class Conta {  
    protected int numero;  
    protected Cliente titular;  
    protected double saldo;  
  
    public void saca(double valor) {  
        saldo = saldo - valor;  
    }  
  
    public void deposita(double valor) {  
        saldo = saldo + valor;  
    }  
  
    public void transfere(double valor, Conta destino) {  
        this.saca(valor);  
        destino.deposita(valor);  
    }  
  
    public abstract boolean validaConta();  
}
```

```
public class ContaCorrente extends Conta {  
  
    private double limite;  
  
    @Override  
    public boolean validaConta() {  
        if (numero > 100000)  
            return false;  
        else  
            return true;  
    }  
}
```

```
public class ContaPoupanca extends Conta {  
  
    private double juros;  
  
    @Override  
    public boolean validaConta() {  
        if (numero <= 100000)  
            return false;  
        else  
            return true;  
    }  
}
```

Programação Orientada a Objetos

Exemplo Banco 1 (Java)

```
public class CadastraContas {  
    public static void main(String[] args) {  
  
        Conta conta = new Conta();      X  
  
        ContaCorrente cc1 = new ContaCorrente();    ✓  
        ContaPoupanca cp1 = new ContaPoupanca();    ✓  
  
        Conta cc2 = new ContaCorrente();    ✓  
        Conta cp2 = new ContaPoupanca();    ✓  
  
        cc1.deposita(300);  
        cc1.transfere(cc2);  
    }  
}
```

Programação Orientada a Objetos

Parte Prática: Exemplo Banco 1 (Java) no BlueJ ou Eclipse.

Programação Orientada a Objetos

- A anotação **@Override** diz ao compilador Java que queremos sobrescrever um método da superclasse. Embora seja desnecessário usar **@Override** sempre que quisermos implementá-lo em um processo, recomendamos usá-lo porque podemos cometer erros ao criar os métodos.
- Para superar o erro, usamos **@Override** acima do nome do método nas classes filhas que informa ao compilador que queremos sobrescrever o método.
 - Se cometer algum erro, o compilador irá gerar um erro.

Programação Orientada a Objetos

Interfaces

- Conceito inspirado nas interfaces do hardware, por exemplo, na figuras vemos as cabos USB 2 e USB 3 de USB tipo A que é o mais comum em PCs e notebooks.

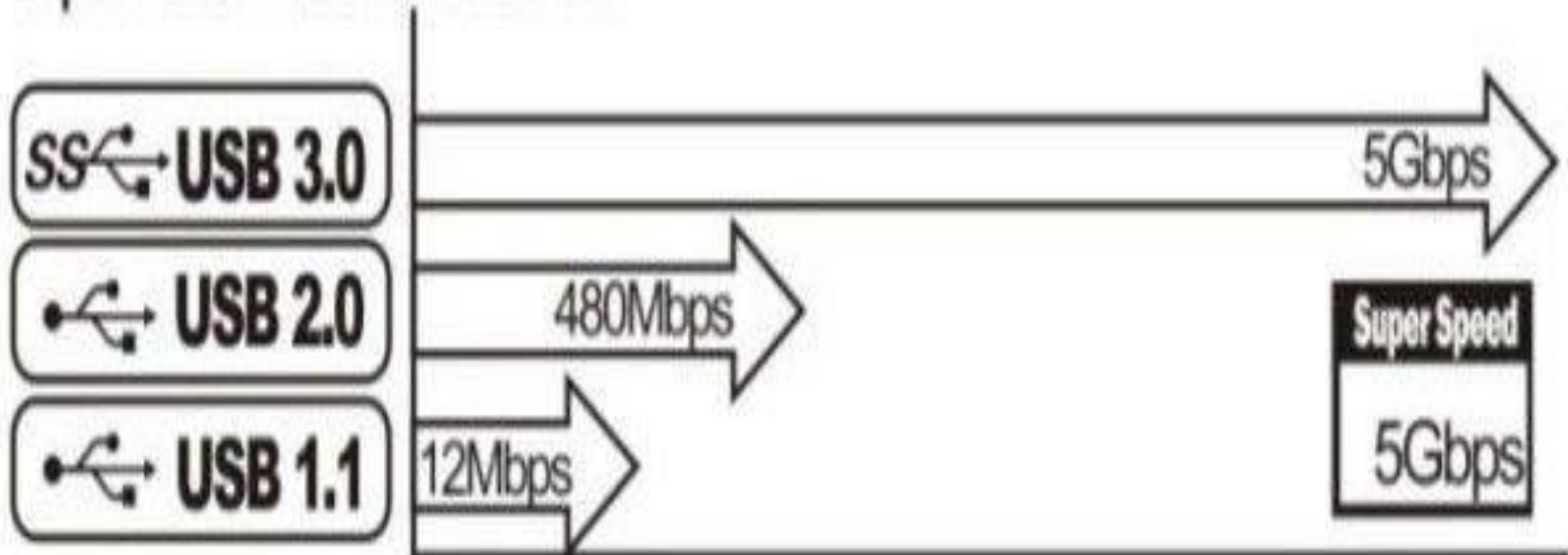


Programação Orientada a Objetos

Interfaces

- É possível ter mais de um tipo de interface, mas com características diferentes, por exemplo, nas portas USB temos diferenças no atributo velocidade.

Speed Difference:



Programação Orientada a Objetos

Interfaces

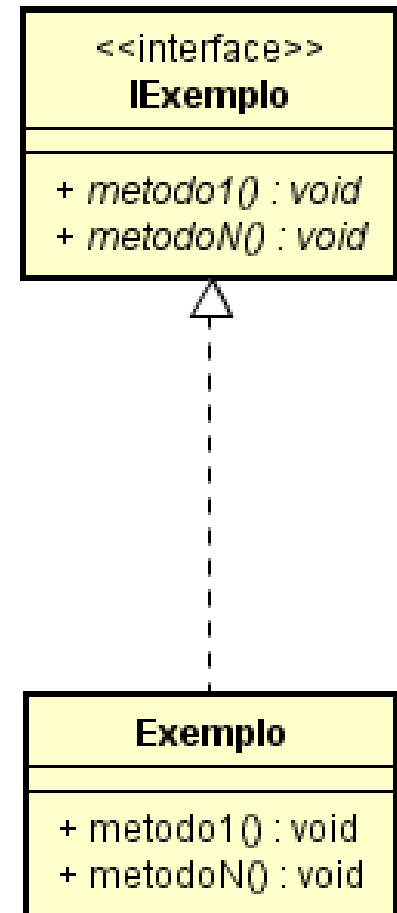
- Uma interface é **como se fosse** uma classe 100% abstrata, ou seja, uma coletânea de métodos sem implementação;
- Uma classe pode ter uma única superclasse (herança simples), mas pode implementar várias interfaces ... através deste recurso que Java consegue se utilizar dos benefícios da herança múltipla;
- Uma interface é um contrato que define um conjunto de métodos públicos vazios que **devem** ser codificados nas subclasses que implementarem a interface;
- Uma interface não pode definir métodos construtores.

Programação Orientada a Objetos

Interfaces

```
public interface IExemplo {  
    public void metodo1();  
    public void metodoN();  
}
```

```
public class Exemplo implements IExemplo {  
    public void metodo1() {  
        System.out.println("Executando metodo1( )...");  
    }  
  
    public void metodoN() {  
        System.out.println("Executando metodoN( )...");  
    }  
}
```



Programação Orientada a Objetos

Exemplo Banco 2 (continuando)

Voltando ao exemplo das Contas bancárias..

Alguns bens dos clientes são tributáveis e outros não: a poupança não é tributável mas a conta corrente, o seguro, as aplicações e outros bens, são tributáveis.

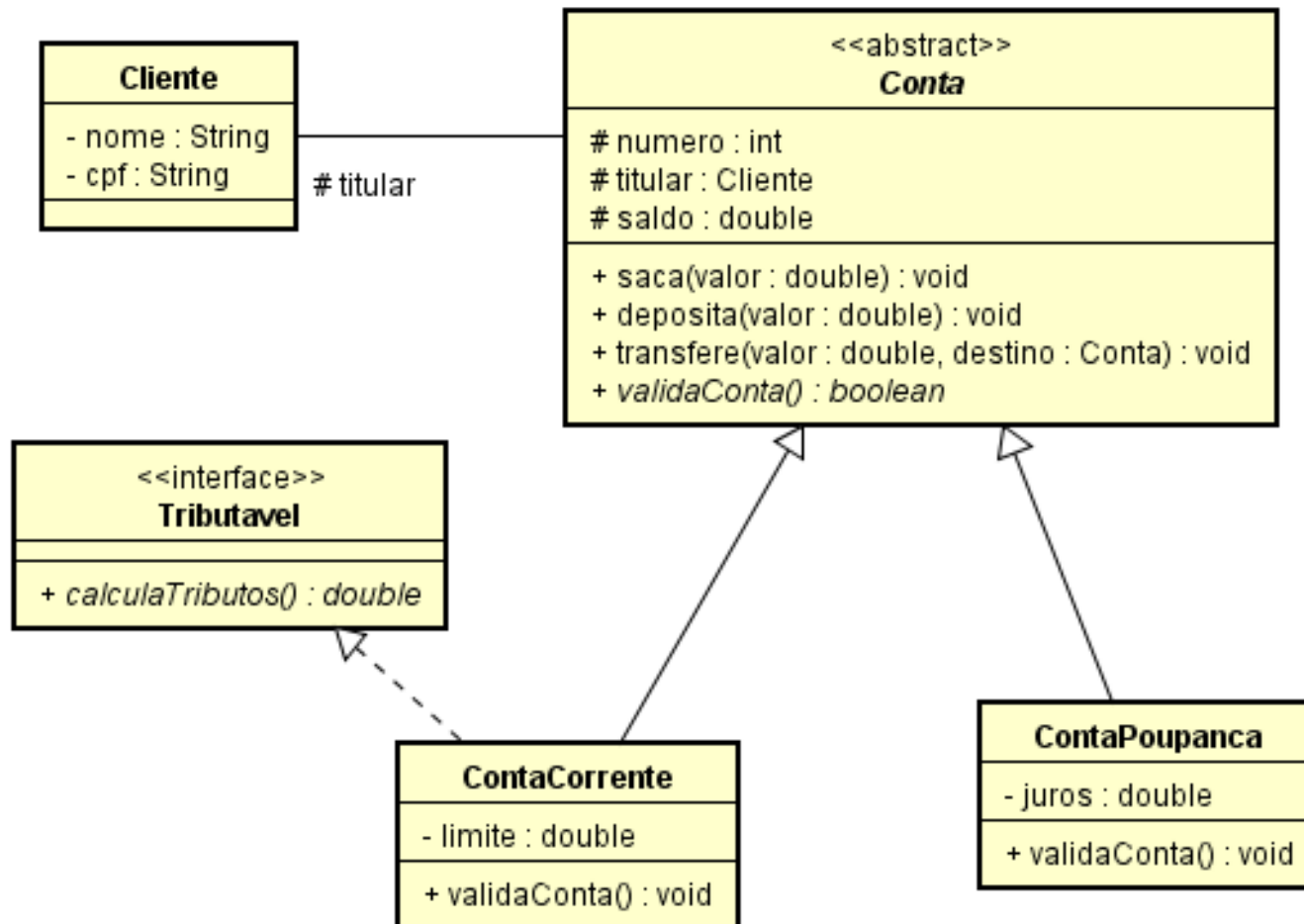
Na conta corrente, o tributo é 1% do saldo, já no seguro é um valor fixo.

Criaremos uma **interface Tributável** que tenha um método para calcular tributo que devolve o valor do tributo (double).

Podemos dizer que todos que precisarem ser Tributáveis devem saber calcular o tributo, ou seja, todos que implementarem essa interface tem a obrigação de fazer o cálculo do tributo.

Programação Orientada a Objetos

Exemplo (UML)



Programação Orientada a Objetos

Exemplo Banco 2 (Java)

```
public abstract class Conta {  
    protected int numero;  
    protected Cliente titular;  
    protected double saldo;  
  
    public void saca(double valor) {  
        saldo = saldo - valor;  
    }  
  
    public void deposita(double valor) {  
        saldo = saldo + valor;  
    }  
  
    public void transfere(double valor, Conta destino) {  
        this.saca(valor);  
        destino.deposita(valor);  
    }  
  
    public abstract boolean validaConta();  
}
```

```
public interface Tributavel {  
    public abstract double calculaTributos();  
}  
  
public class ContaCorrente extends Conta  
    implements Tributavel {  
    private double limite;  
  
    @Override  
    public boolean validaConta() {  
        if (numero > 100000)  
            return false;  
        else  
            return true;  
    }  
  
    @Override  
    public double calculaTributos() {  
        return saldo * 0.01;  
    }  
}
```

Programação Orientada a Objetos

Parte Prática: Exemplo Banco 2 (Java) no BlueJ ou Eclipse.

Programação Orientada a Objetos

Classes Abstratas x Interfaces

- Ambos são projetos de classes que não podem ser instanciados;
- Uma interface é uma classe 100% abstrata (que não contém nenhum método implementado, apenas assinaturas);
- Uma classe pode herdar uma única classe abstrata mas pode implementar várias interfaces;
- Uma classe abstrata pode conter atributos de instância, já uma interface pode conter apenas constantes estáticas;
- Em uma interface os métodos são implicitamente públicos e abstratos.
- Ambas exigem a implementação dos métodos abstratos nas subclasses;